Amendments to the Specification

Please replace the paragraph beginning on page 25, line 1, with the rewritten paragraph below.

Fig. 8 illustrates a Swing-type API 74 (region enclosed within the dotted lines). In Fig. 8, Button 30 is derived from the Component class 32. Also, the Peer mechanism links the Button object 30 with the graphics resources used to create and manipulate its graphic image 26, which is contained within Frame ~~90~~41. However, in this case, the JButtonPeer 82 (and the JComponentPeer 84, from which it is derived) are lightweight Peers, written entirely in Java and having no dependence on the windowing resources of the operating system. Instead, the JComponentPeer 84 intercepts key query commands related to the location and event status of the Button object 30 and routes them to a JButtonProxy object 88. In the present embodiment, the following methods of the Swing object are intercepted and redirected by the proxy object:

isShowing();
getLocationOnScreen();
1getGraphics();
proxyRequestFocus();
requestFocus();
getInputContext();
getLocale();
nextFocus();
createImage();
getParent();

A1

As a result, the JButtonProxy 88 responds to key events, such as focus, mouse, sizing, etc., directed to the original AWT-based control. JButtonProxy 88 is a subclass of the Swing JComponent class and, indirectly, of the JComponent class, and inherits methods and properties thereof. However, Java allows for customization (commonly referred to as "overriding") of inherited methods. In particular, the getParent() method of the JButtonProxy identifies as its parent the Frame ~~90~~41 of the target Button 30. Thus, the JButton "thinks" it belongs in the target's Frame, while the Frame knows nothing of the existence of the Swing component. Furthermore, when the Button 30 is asked to draw itself, the call to the drawing method of legacy AWT object is redirected by lightweight Peer 84 to JButtonProxy 88, which accesses the drawing methods of the JButton class 48. This invokes the platform-independent methods of Swing to paint the region of the screen originally allocated for the Button 30 in the AWT-based application.